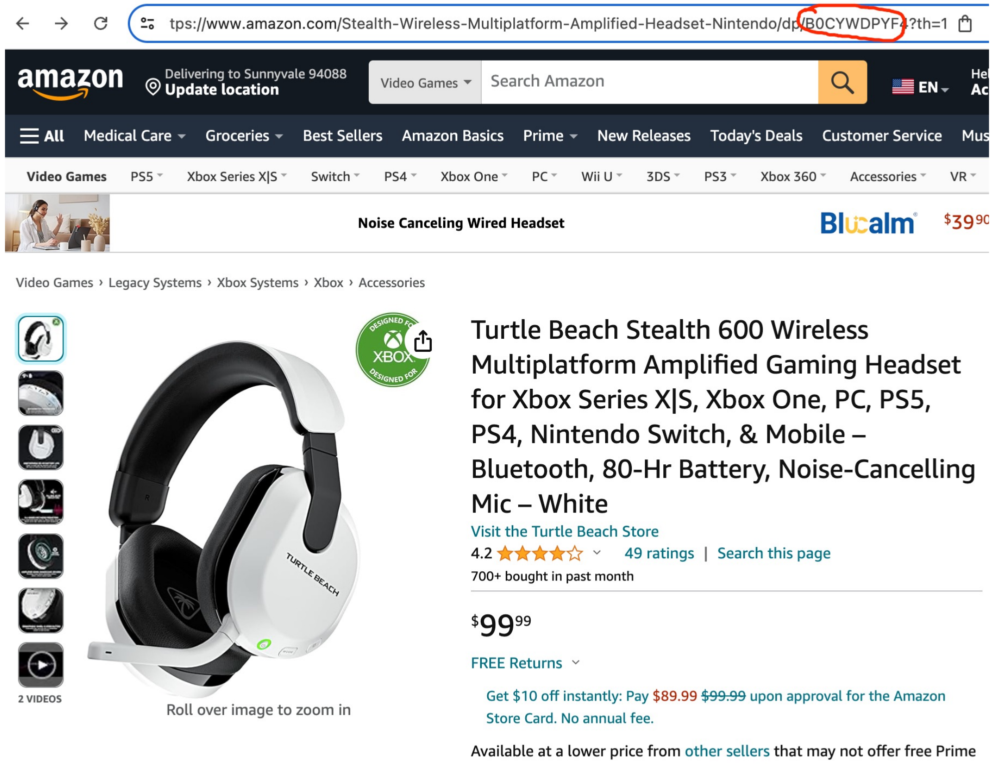
# Language Models as Semantic Indexers

Bowen Jin[1], Hansi Zeng[2], Guoyin Wang[3], Xiusi Chen[4], Tianxin Wei[1], Ruirui Li[3], Zhengyang Wang[3], Zheng Li[3], Yang Li[3], Hanqing Lu[3], Suhang Wang[5], Jiawei Han[1], Xianfeng Tang[3]

[1]UIUC, [2]UMass, [3]Amazon, [4]UCLA, [5]Penn Sate

# Introduction

- **Background**
  - Unique IDs are assigned to documents for indexing and retrieval.
    - E-commerce products have distinctive product IDs.
    - Web pages are linked to specific URLs.




# Introduction

- **Background**
  - Unique IDs are assigned to documents for indexing and retrieval.
    - E-commerce products have distinctive product IDs.
    - Web pages are linked to specific URLs.

  - However, these IDs are often randomly assigned and lack the assurance of the content information of items and documents.


all/dp/B08QJLXZ45?th=1


ted/dp/B000067R1I

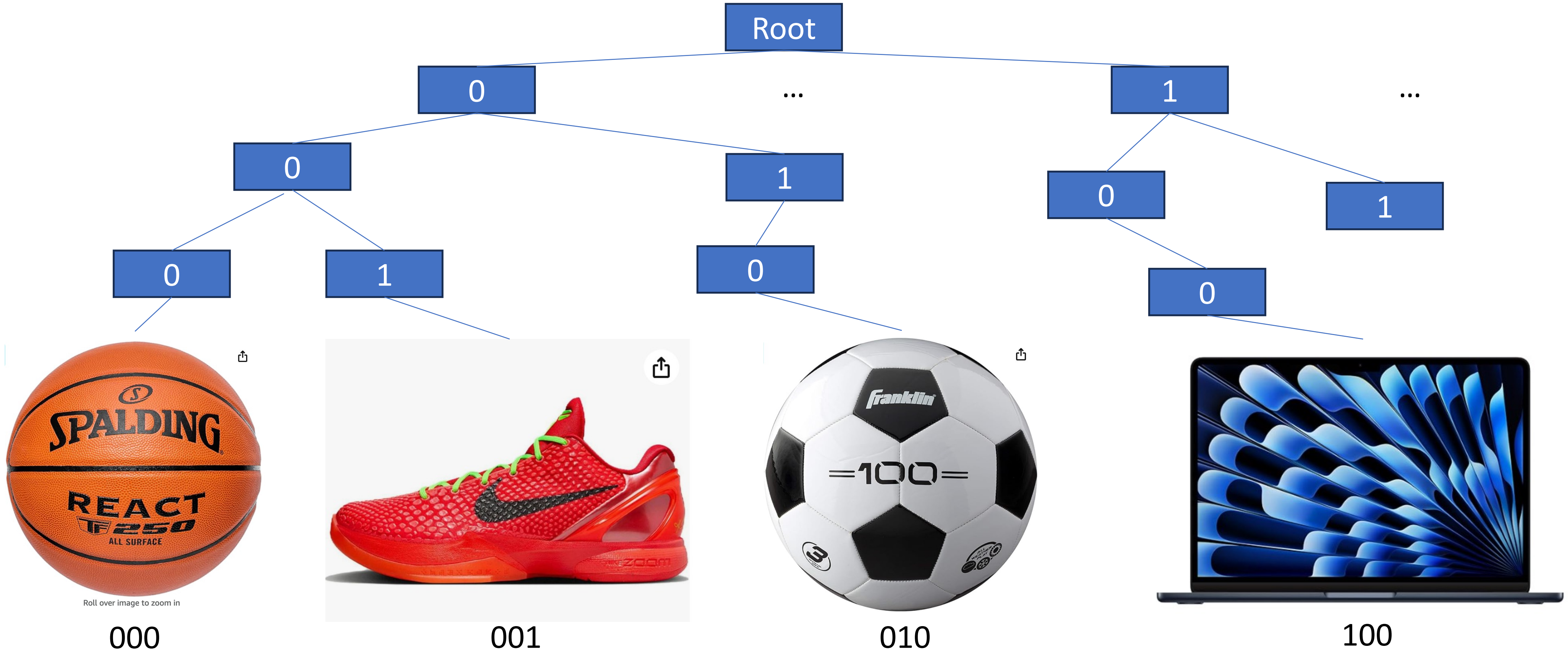
op/dp/B0CX23V2ZK

# Introduction

- **Background**
  - However, these IDs are often randomly assigned and lack the assurance of the content information of items and documents.
  - This hinders the effective understanding, indexing and searching based solely on IDs.

# Introduction

- **Semantic ID**
  - A sequence of discrete ID numbers that captures the semantic meaning of a document.
  - The objective is to ensure that the initial set of semantic IDs captures the coarse-grained document semantics while the subsequent IDs delve into the details of its content in a hierarchical structure.



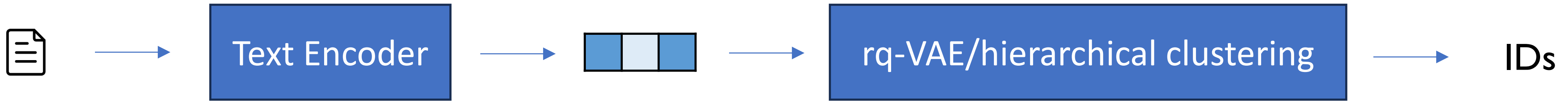
# Introduction

- **How can we assign semantic IDs to documents?**
  - A straightforward way is to use the category information or external hierarchy.
  - However, such external information not always exist.
  - In many cases, we only have text associated with each document.

- **Problem definition (Learning semantic IDs with text self-supervision)**
  - Input:
    - A corpus of documents with texts.
  - Output:
    - Semantic ID for each document in the input corpus.

# Existing works

- **Two-step methodology**
  - Step 1: procure embeddings for documents with off-the-shelf text encoders.
  - Step 2: specific techniques, e.g., rq-VAE or hierarchical clustering to derive IDs.




- **Limitations**
  - Inherent mismatch between the distribution of the embeddings in the latent space generated by encoder and the expected distribution for semantic indexing.
  - Each step of this process introduces potential information loss.

# Our solution: LMIndexer

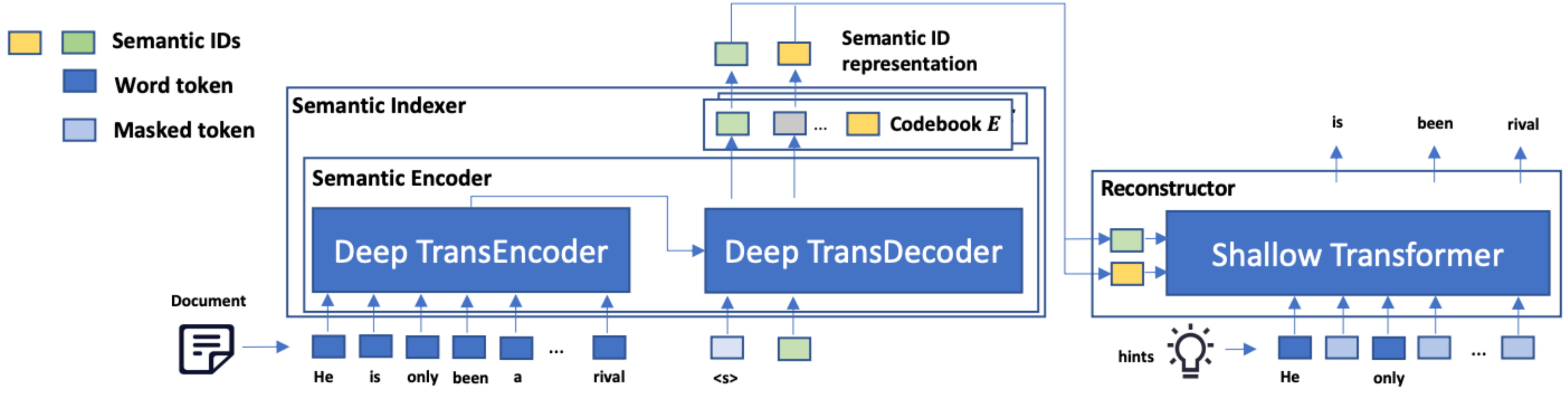
- **Single step: Learn a language model as a semantic indexer**


- **This is non-trivial given that**
  - **We do not have any ID supervision:** Let's use the self-supervision from text itself to learn the IDs.
  - **The IDs are discrete rather than continuous (hard to optimize).**

# Our solution: LMIndexer

- **Single step: Learn a language model as a semantic indexer**


- **Learning Semantic IDs with Sequential Discrete Auto-reconstruction**
  - Self-supervision learning to alleviate the lack of ID supervision.
  - Learn the semantic IDs with sequential discrete representations.



# Our solution: LMIndexer

- **Learning Semantic IDs as Neural Sequential Discrete Representations**
  - We adopt an encoder-decoder Transformer (T5) as the base model.
  - $c_d^i$ denote the semantic ID of the document $d$ at the position $i$.
  - We first learn the continuous representation at position $t$ as

$$\boldsymbol{h}_d^t = \text{SemEnc}_\theta(d, c_d^{<t}) = \text{TransDecoder}(\text{TransEncoder}(d), c_d^{<t}). \quad (1)$$

  - The continuous representation $h_d^t$ is then projected to a discrete representation by

$$P_s(c_d^t = j | c_d^{<t}, d) = \text{Softmax}_{\boldsymbol{e}_j^t \in \boldsymbol{E}^t}(\boldsymbol{h}_d^t \cdot \boldsymbol{e}_j^t),$$
$$c_d^t = \text{argmax}_j P_s(c_d^t = j | c_d^{<t}, d).$$

# Our solution: LMIndexer

- **Reconstructing Document with Sequential Discrete Semantic ID Embeddings**
  - Basically, we use the semantic IDs $\boldsymbol{c}_d$ to reconstruct the original document $d$.
  - If this can be well-performed, this means that $\boldsymbol{c}_d$ contains enough semantic information.
  - However, solely based on $\boldsymbol{c}_d$ is difficult. We consider provide some hints $d_h$.

$$\mathcal{L}_{\text{recon}} = -\sum_{d} \sum_{w \in d \backslash d_{\text{h}}} \log P_{\text{recon}}(w|\boldsymbol{c}_d, d_{\text{h}}).$$

  - We adopt a shallow Transformer as the reconstructor.

$$\boldsymbol{z}_w = \text{Recon}_{\phi}(\boldsymbol{c}_d, \boldsymbol{d}_{\text{h}}) = \sum_{t} \text{Trans}(q = \boldsymbol{c}_d^t, k = \boldsymbol{d}_{\text{h}}, v = \boldsymbol{d}_{\text{h}})$$

$$P_{\text{recon}}(w|\boldsymbol{c}_d, d_{\text{h}}) = \text{softmax}(\boldsymbol{W}\boldsymbol{z}_w)$$

# Our solution: LMIndexer

- **Reconstructing Document with Sequential Discrete Semantic ID Embeddings**
  - However, directly adopting the reconstruction objective with $c_d$ as input to the reconstructor will not optimize the semantic encoder.
  - The codebook look-up is a hard/discrete operation.
  - To this end, we propose to approximate the argmax operation with

$$\hat{\boldsymbol{c}}_d^t = \begin{cases} \arg\max_{\boldsymbol{e}_j^t \in \boldsymbol{E}^t} \boldsymbol{h}_d^t \cdot \boldsymbol{e}_j^t & \text{forward pass.} \\ \sum_{\boldsymbol{e}_j^t \in \boldsymbol{E}^t} \frac{\exp(\boldsymbol{h}_d^t \cdot \boldsymbol{e}_j^t)}{\sum_{\boldsymbol{e}_j^t \in \boldsymbol{E}^t} \exp(\boldsymbol{h}_d^t \cdot \boldsymbol{e}_j^t)} \boldsymbol{e}_j^t & \text{backward pass.} \end{cases}$$

  - In our implementation, we achieve this by adopting the "stop gradient" operation.
  - The final reconstruction loss is

$$\boldsymbol{z}_w = \text{Recon}_\phi(\hat{\boldsymbol{c}}_d^t, \boldsymbol{d}_\text{h}) = \sum_t \text{Trans}(q = \hat{\boldsymbol{c}}_d^t, k = \boldsymbol{d}_\text{h}, v = \boldsymbol{d}_\text{h})$$

# Our solution: LMIndexer

- **Training self-supervised semantic indexer**
  - Progressive training: IDs have dependencies.

$$\mathcal{L}^t_{\text{recon}} = -\sum_{d} \sum_{w \in d \backslash d^t_{\text{h}}} \log P_{\text{recon}}(w | \boldsymbol{c}^{\leqslant t}_d, d^t_{\text{h}}).$$

  - Contrastive loss: promote distinction between documents that shared the same prefix.

$$\mathcal{L}^t_{\text{contrastive}} = -\sum_{d} \log \frac{\exp(\boldsymbol{h}^t_d \cdot \boldsymbol{h}^t_d)}{\exp(\boldsymbol{h}^t_d \cdot \boldsymbol{h}^t_d) + \sum_{c^{<t}_{d'} = c^{<t}_d} \exp(\boldsymbol{h}^t_d \cdot \boldsymbol{h}^t_{d'})}.$$

  - Commitment loss: force the semantic indexer to remember the previous learnt IDs.

$$\mathcal{L}^t_{\text{commitment}} = -\sum_{d} \sum_{j<t} \log P_s(c^j_d | d, c^{<j}_d).$$

# Our solution: LMIndexer

- **Training self-supervised semantic indexer**
  - Final loss: a combination of the three losses.

$$\min_{\theta,\phi,\boldsymbol{E}^t} \mathcal{L}^t = \mathcal{L}^t_{\text{recon}} + \mathcal{L}^t_{\text{contrastive}} + \mathcal{L}^t_{\text{commitment}}.$$

  - Reconstructor collapse: constructor is performing badly and misguides the semantic indexer.

$$\min_{\phi} \mathcal{L}^0_{\text{recon}} = -\sum_{d} \sum_{w \in d \backslash d^0_{\text{h}}} \log P_{\text{recon}}(w | d^0_{\text{h}}).$$

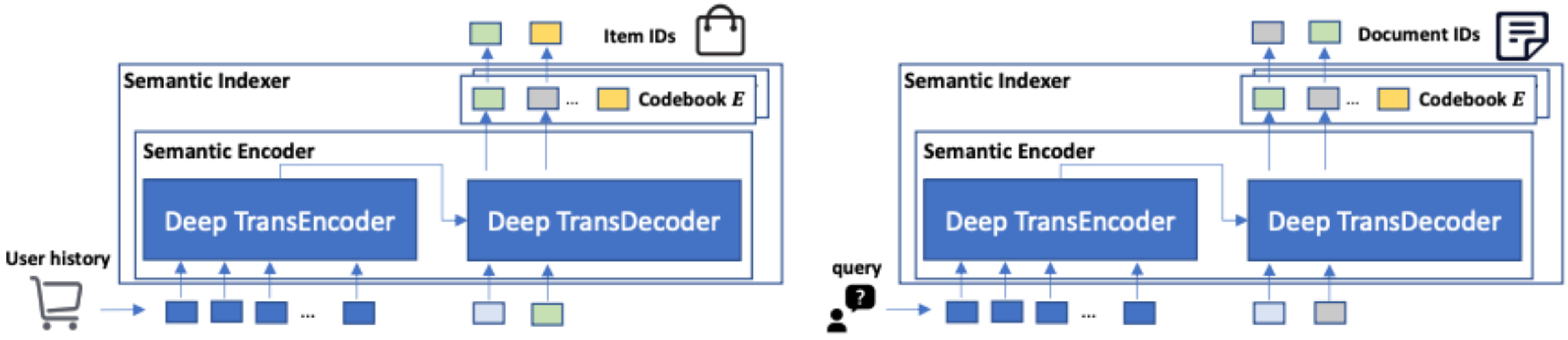
  - Posterior collapse: information provided by the semantic indexer is weak and noisy for the reconstructor.

$$\min_{\theta,\phi} \mathcal{L}^t, \quad \boldsymbol{z}_w = \text{Recon}_{\phi}(\boldsymbol{c}^{<t}_d, \boldsymbol{h}^t_d, \boldsymbol{d}^t_{\text{h}})$$

# Our solution: LMIndexer

- **Finetuning semantic indexer on downstream tasks**
  - Downstream tasks which take text as input and expect document IDs as output.
  - E.g., recommendation (user history text as input, next item ID as output)
  - E.g., retrieval (query as input and document ID as output)

$$\mathcal{L}_{\text{downstream}} = - \sum_{(q, c_d) \in \mathcal{D}} \sum_{j \leqslant T} \log P_s(c_d^j | q, c_d^{<j}).$$



(a) Recommendation (b) Retrieval

# Experiments

- **Datasets:**
  - Amazon
    - Beauty, Sports, Toys
  - Wiki
    - NQ320k
  - Web
    - MACRO 1M
    - TREC_DL 1M

| Dataset | # Items | # Users | # Rec history (train/dev/test) |
|---|---|---|---|
| Amazon-Beauty | 12,101 | 22,363 | 111,815 / 22,363 / 22,363 |
| Amazon-Sports | 18,357 | 35,598 | 177,990 / 35,598 / 35,598 |
| Amazon-Toys | 11,924 | 19,412 | 97,060 / 19,412 / 19,412 |

| Dataset | # Documents | # Query (train/test) | # Search labels (train/test) |
|---|---|---|---|
| NQ320k | 109,739 | 307,373 / 7,830 | 307,373 / 7,830 |
| MACRO 1M | 1,000,000 | 502,939 / 6,980 | 532,751 / 7437 |
| TREC-DL 1M | 1,000,000 | 502,939 / 93 | 532,751 / 1,069 |

- **Downstream tasks**
  - Recommendation
  - Retrieval

# Experiments: Learning Self-supervised Semantic ID

- Semantic ID Analysis (quantitative results)

*Table 1.* ID quantitative study (AMI) on Amazon datasets.

| Model | Beauty | Sports | Toys |
|---|---|---|---|
| rq-VAE indexer (BERT) | 0.2654 | 0.2774 | 0.3154 |
| HC indexer (BERT) | 0.2428 | 0.2387 | 0.2729 |
| rq-VAE indexer (In-domain SimCSE) | 0.3100 | 0.2695 | 0.3126 |
| HC indexer (In-domain SimCSE) | 0.2771 | 0.2622 | 0.2968 |
| LMINDEXER | **0.3563** | **0.4163** | **0.3536** |

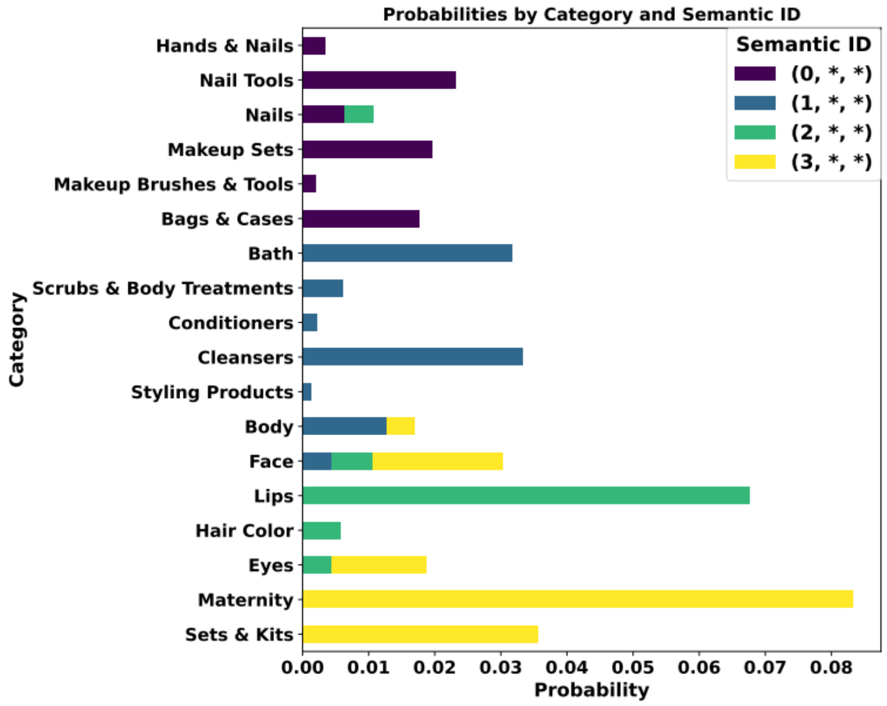
*Table 13.* Human evaluation of semantic ID quality.

| Model | Accuracy |
|---|---|
| rq-VAE indexer | 0.7375 |
| HC indexer | 0.5375 |
| LMINDEXER | 0.7750 |

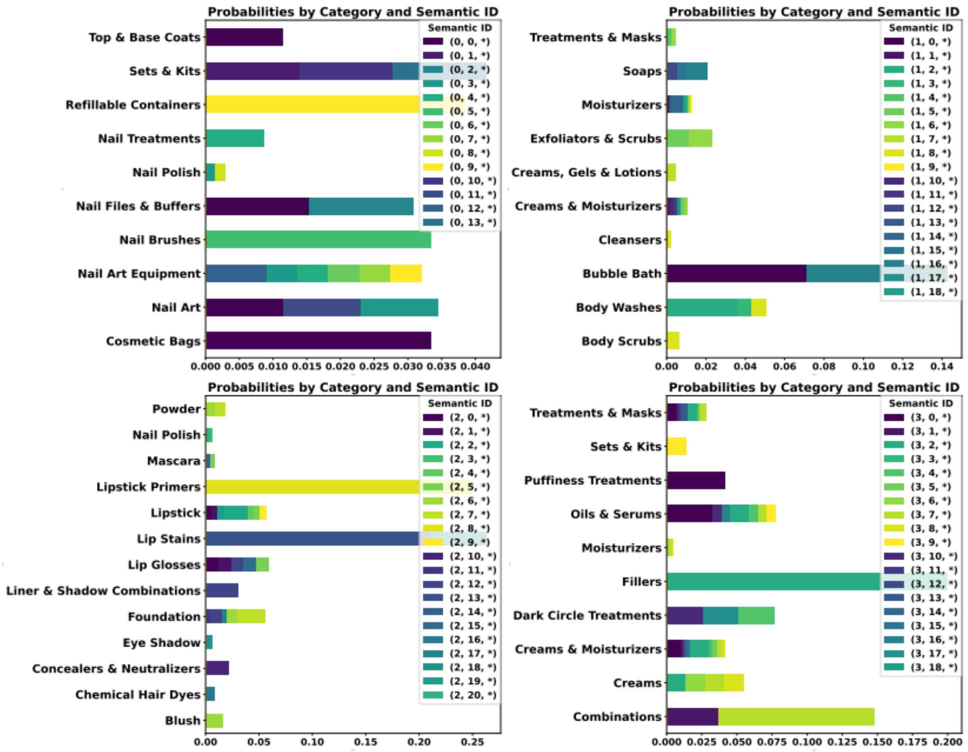
- LMIndexer outperforms baselines consistently, which demonstrates that the IDs learned by LMIndexer are more semantic-indicative.

# Experiments: Learning Self-supervised Semantic ID

- Semantic ID Analysis (qualitative results)




(a) The ground-truth category distribution for items in the Amazon-Beauty dataset is colored by the value of first ID $c^1$.

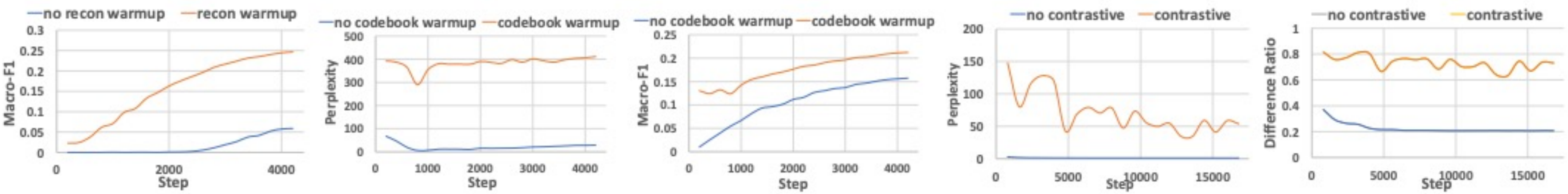



(b) The category distributions for items having the Semantic ID as $(c^1, *, *)$, where $c^1 \in \{0, 1, 2, 3\}$. The categories are colored based on the second semantic token $c^2$.

- $c^1$ captures the coarse-grained category.
- $c^2$ further categorizes into fine-grained categories.

# Experiments: Learning Self-supervised Semantic ID

- Training study




(a) RC: Macro-F1 (b) PC: perplexity (c) PC: Macro-F1 (d) CL: perplexity (e) CL: diff ratio

*Table 2.* Ablation study of commitment loss.

| Dataset | Sports | Toys | Beauty |
|---|---|---|---|
| w. commitment loss | 305.39 | 280.30 | 287.01 |
| w/o commitment loss | 147.10 | 211.60 | 261.04 |

- Reconstructor collapse and posterior collapse exist without proper warm up operations.
- Contrastive loss can facilitate ID distinction and diversity.
- Commitment loss can force the semantic indexer remember the previous learned IDs.

# Experiments: Downstream Tasks

- Sequential Recommendation

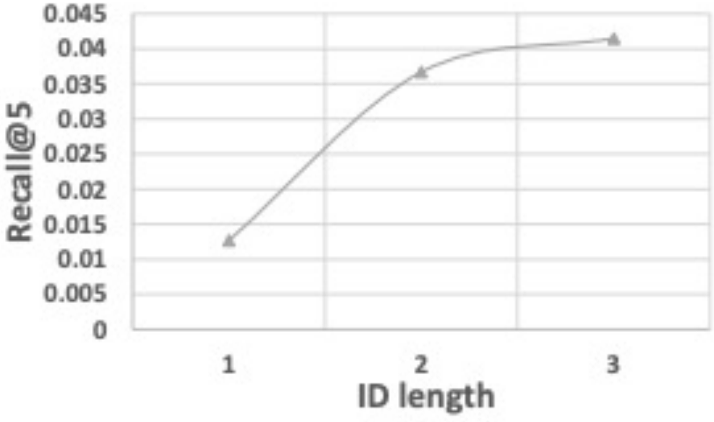
Table 3. Next item recommendation.

| | Amazon-Beauty | | Amazon-Sports | | Amazon-Toys | |
|---|---|---|---|---|---|---|
| Model | Recall@5 | NDCG@5 | Recall@5 | NDCG@5 | Recall@5 | NDCG@5 |
| HGN | 0.0325 | 0.0206 | 0.0189 | 0.0120 | 0.0321 | 0.0221 |
| GRU4Rec | 0.0164 | 0.0099 | 0.0129 | 0.0086 | 0.0097 | 0.0059 |
| BERT4Rec | 0.0203 | 0.0124 | 0.0115 | 0.0075 | 0.0116 | 0.0071 |
| FDSA | 0.0267 | 0.0163 | 0.0182 | 0.0122 | 0.0228 | 0.0140 |
| rq-VAE indexer | 0.0136 | 0.0086 | 0.0067 | 0.0040 | 0.0084 | 0.0055 |
| HC indexer | 0.0129 | 0.0078 | 0.0076 | 0.0050 | 0.0082 | 0.0054 |
| LMINDEXER | **0.0415** | **0.0262** | **0.0222** | **0.0142** | **0.0404** | **0.0268** |

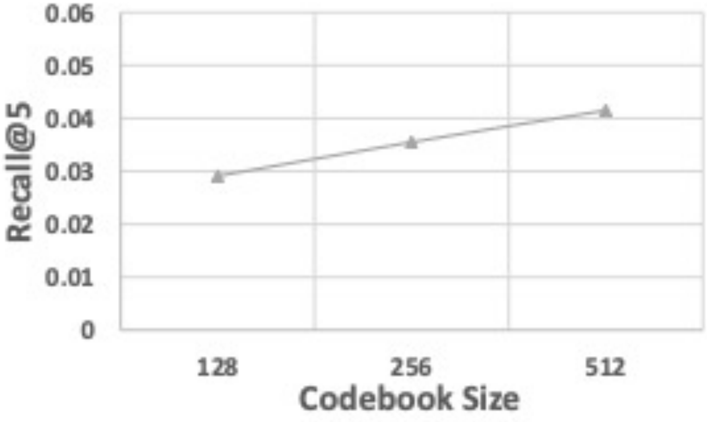
- LMIndexer outperforms the competitive baseline methods consistently and significantly.

# Experiments: Downstream Tasks

- Sequential Recommendation




(a) ID length




(b) Codebook size

- The model performance increases as the semantic ID length or codebook size increases.

# Experiments: Downstream Tasks

- Product Search

*Table 4.* Product search.

| | Amazon-Beauty | | Amazon-Sports | | Amazon-Toys | |
|---|---|---|---|---|---|---|
| Model | NDCG@5 | MAP@5 | NDCG@5 | MAP@5 | NDCG@5 | MAP@5 |
| bm25 | 0.2490 | 0.2152 | 0.1898 | 0.1581 | 0.2085 | 0.1760 |
| Dual Encoder | 0.2565 | 0.2096 | 0.2556 | 0.2223 | 0.2805 | 0.2420 |
| SEAL | 0.1271 | 0.1050 | 0.2011 | 0.1739 | 0.1035 | 0.0843 |
| rq-VAE indexer | 0.2710 | 0.2469 | 0.2606 | 0.2354 | 0.2511 | 0.2287 |
| HC indexer | 0.2172 | 0.1959 | 0.1979 | 0.1812 | 0.2379 | 0.2156 |
| LMINDEXER | **0.3187** | **0.2888** | **0.2870** | **0.2607** | **0.2865** | **0.2592** |

- LMIndexer outperforms the competitive baseline methods consistently and significantly.

# Experiments: Downstream Tasks

- Product Search

*Table 5.* Study of the number of layers in reconstructor on Amazon-Beauty dataset. AMI, Recall@5, and NDCG@5 are used as metrics for ID quality study, recommendation, and retrieval.

| Model | ID quality | Recommendation | Retrieval |
|---|---|---|---|
| LMINDEXER (Recon 1 layer) | **0.3563** | **0.0415** | **0.3187** |
| Recon 2 layers | 0.2390 | 0.0284 | 0.2528 |
| Recon 3 layers | 0.1679 | 0.0281 | 0.2522 |

- As the reconstructor layer increases, the quality of the semantic indexer and its generated IDs decreases.

# Experiments: Downstream Tasks

- Document retrieval

*Table 6.* Document retrieval.

| | NQ320k | | TREC-DL 1M | | MACRO 1M |
|---|---|---|---|---|---|
| Model | Recall@1 | Recall@10 | Recall@10 | NDCG@10 | MRR@10 |
| bm25 | 0.2970 | 0.6030 | 0.2756 | 0.2995 | 0.3144 |
| Dual Encoder | 0.5360 | 0.8300 | 0.3612 | 0.3941 | **0.5561** |
| SEAL | 0.5990 | 0.8120 | - | - | - |
| rq-VAE indexer | *0.6480* | *0.8322* | 0.4199 | *0.4579* | 0.5159 |
| HC indexer | 0.6439 | 0.8213 | *0.4265* | 0.4571 | 0.5126 |
| LMINDEXER | **0.6631** | **0.8589** | **0.4519** | **0.4695** | *0.5485* |

- LMIndexer outperforms the competitive baseline methods consistently and significantly.

# Conclusion

- In this work, we explore language models as semantic indexers and learn the IDs with only one step.
- We propose a neural sequential discrete auto-reconstruction pipeline to train the semantic indexer with self-supervision.
- We conduct experiments on real-world datasets from both e-commerce and web and demonstrate the effectiveness of our method on both recommendation and retrieval downstream tasks.

# Thank You !

Subscribe and learn
more about our works!

Code, can be found here
https://github.com/PeterGriffinJin/LMIndexer!